Упутство за Лпрс асемблер

Преглед

Ово је упутство за коришћење асемблерског преводиоца за Лпрс процесорско језгро, тј. процесорско језгро које се ради на вежбама из предмета Логичко пројектовање рачунарских система. Асемблерски преводилац је направљен на одсеку за рачунарску технику и рачунарске комуникације Факултета Техничких Наука и користе се у образовне сврхе. Молимо да све грешке које приметите пријавите на адресу sppuv1@rt-rk.com.

Асемблер се покреће из командне линије. На улазу прима једну текстуалну датотеку која садржи асемблерски код. Над улазном датотеком се најпре обавља Це претпроцесирање. Дакле, сва претпроцесорска функционалност Це језика доступна је и у овом асемблеру, пре свега укључивање заглавља и писање макроа.

Излаз из асемблера су две текстуалне датотека. Једна датотека садржи извршиви машински код, а друга датотека садржи податке. Није потребно никакво повезивање, већ се текст из сваке датотеке треба копирати на одговарајуће место у VHDL датотекама процесора.

Командна линија

Формат командне линије изгледа овако:

```
lprsasm [options] <input filename> [options]
```

Следи листа опција:

Опције општег типа	

`-h`
`-help`
`-?`
	Исписује сажети опис командне линије на стандардни излаз.

`-v`	Исписује податке о верзији алата.
Асемблерске опције	
`-o<file>`	Специфицира назив излазних датотека. Једна ће имати екстензију „.o.txt“, а друга екстензију „.o.dat“. Подразумеван назив излазних датотека је назив улазне датотеке.
`-I<path>`	Специфицира додатну путању на којој ће бити тражена заглавља која се претпроцесорском директивом укључују. Заглавља ће свакако бити тражена на путањи на којој се налази улазна датотека.
`-D<symbol[=value]>`	Дефинише претпроцесорски симбол. Симболу се може придружити и вредност.

Претпроцесирање

Као први корак асемблирања позива се Це претпроцесор (заснова GNU C предњем делу) над улазном датотеком. Остатак асемблирања се обавља над претпроцесираном датотеком. На овај начин асемблерски преводилац омогућава коришћење макроа, условно превођење, именовање нумеричких литерала, укључивање заглавља и све остало што Це претпроцесор нуди. Из овога такође произилази начин писања коментара у коду: линијског коментара коришћењем //, или блоковског коментара. коришћењем /* */ конструкције.

У дефиницији Це претпроцесорских макроа не може се налазити нови ред, и то представља значајно ограничење за писање сложених, вишелинијских макроа за асемблерски језик. Због тога је уведен предефинисани симбол __NEW_LINE__ који ће од стране асемблерског преводиоца бити третиран као ознака новог реда.

Асемблерске директиве

Директиве увек почињу тачком и морају бити у једном реду.

Следи листа директива:

Директива за текст секцију

```
.text
```

Означава почетак секције у којој се могу налазити инструкције. У програму може постојати само једна текст секција. Адресе инструкција у текст секцији крећу од нуле (скреће се пажња на чињеницу да се излаз из асемблера директно пуни у меморију, тј. да нема повезивања).

Директива за секцију података

```
.data
```

Означава почетак секције у којој се могу налазити подаци. У програму може постојати само једна секција података. У сваком реду ове секције може се налазити по један израз (у складу са описом израза из поглавља Опште конвенције). Вредност израза представља 15битни садржај меморије. Адресе у овој секцији такође крећу од нуле, у складу са чињеницом да нема повезивања и да је адресни простор за инструкције и податке одвојен.

Опште конвенције

Лабеле (Симболи)

Лабеле су на почетку реда и завршавају се двотачком. Не може бити више лабела са истим именом. Као пример дат је програмски код бесконачне петље, тј. инструкција скаче на себе саму. Ову конструкцију је згодно ставити на крај програма.

```
Labela:
  jmp Lebela
```

Изрази

На месту непосредних операнада Лпрс асемблер прихвата изразе. У изразима као операнди могу учествовати цели бројеви и називи лабела. Цели бројеви могу бити изражени у декадном бројном систему, што је подразумевано, али и у хексадецималном, ако имају префикс „0x“. У изразу може учествовати највише један назив лабеле. Потребно је да је израз могуће свести или на цео број или на форму: назив лабеле + цео број. У изразу могу учестовати следеће операције: +, -, /, *, и <<.

Примери програма

Следи два примера програма који множе два позитивна броја из меморије.

Пример 1

```
#define ZERO(x) sub x, x, x

.data
  45 // a, adresa 0
  123 // b, adresa 1

.text
Begin: // program racuna a * b
  ZERO(R0) // postavlja R0 na nulu
  ld R1, R0 // R1 - vrednost sa adrese 0
  inc R0, R0
  ld R2, R0 // R2 - vrednost sa adrese 1
  dec R0, R0
  // Stanje: R0: 0, R1: a, R2: b
Loop:
  add R0, R0, R1
  dec R2
  jmpnz Loop
End:
  jmp End// rezultat je u R0
```

Пример 2

```
#define ZERO(x) sub x, x, x

.data
  45 // a, adresa 0
  123 // b, adresa 1

.text
Begin: // program racuna a * b
  ZERO(R0) // postavlja R0 na nulu
  ld R1, R0 // R1 - sa lokacije a:
  inc R3, R0
```

```
  ld R2, R3 // R2 - sa lokacije b:
  // Stanje: R0: 0, R1: a, R2: b, R3: 1
Loop:
  // R2 neparno?, tj. da li je poslednji bit 1?
  and R4, R2, R3
  jmpz NoAdd // ako nije, onda preskoci sabiranje
  add R0, R0, R1
NoAdd:
  add R1, R1, R1 // R1 <- 2*R1
  shr R2, R2
  jmpnz Loop
End:
  jmp End // rezultat je u R0
```

Д

Историја промена

Верзија	Датум	Аутор	Опис
1.0	2017.12.26.	М. Ђукић	Прва верзија
1.1	2018.01.04.	М. Ђукић	Додати примери и мејл адреса за пријаву грешака